A Sampling of Surface Reconstruction Techniques

Huong Quynh Dinh

1 Introduction

The goal of surface reconstruction is to obtain a continuous representation of a surface described by a cloud of points. This problem is often called the *unorganized points problem* because the cloud of points has no connectivity information. This paper surveys the solution techniques for the unorganized points problem. Two closely related formulations of the problem are surface interpolation and approximation. Many reconstruction techniques handle only exact interpolation, while others can vary from exact to approximate surfaces. Exact and approximate surfaces differ in that exact surfaces pass through the data points, while approximate surfaces pass near the data points.

The motivation behind surface reconstruction is to obtain a digital representation of a real world, physical object or phenomenon. Clouds of point data may be obtained from medical scanners (X-rays, MRI), laser range finders (optical, sonar, radar), or vision techniques (correlated viewpoints, voxel carving, stereo range images). Often, additional information on the cloud of points may be available, such as the order in which the data points were sampled, the orientation of the normal vector at each of the points, or the positions of the cameras used in stereo range images. Some surface reconstruction algorithms take into consideration this information, while others tackle the general problem.

Notions which often appear in the surface reconstruction literature include best fit, least error, distance metric, smooth, piecewise, and energy minimizing, among others. These notions help to distinguish the different techniques. One of the primary problems in reconstruction is that a "correct" surface is not defined. That is, we may have a cloud of points that is a discrete representation of an organ, but how can we evaluate a surface reconstructed from the data? How can we decide that we have obtained a good representation of the organ when we have no way to exactly measure the continuous surface of the organ? In addition, the data itself may be noisy. Even if we reconstruct the surface of an object whose surface we know analytically, such as a unit sphere or cube, we are not guaranteed that a reconstruction algorithm that perfectly reconstructs our known object would do as well with any other unknown object. As a result, notions such as least error, distance metrics, smoothness of the surface, and energy minimizing become important metrics with which to evaluate a reconstructed surface. In one scenario, the best reconstruction is one that minimizes the distance between all data points and the reconstructed surface.

This paper compares several of the recent techniques in the universe of surface representation and reconstruction. In particular, more attention is given to the algebraic domain than to the computational geometry domain. There are three primary categories of surface representations:

Parametric The surface is represented by a patch, described by a parametric equation. Multiple patches may be pieced together to form a continuous surface. Examples of parametric representations include B-spline, Bezier, and Coons patches. The basic parametric formulations such as B-splines will not be discussed in this paper. Instead, two examples of reconstruction to parametric representations will be presented. Sections 5 and 6 cover Terazopoulos' thin-plate parametric representation and Gossard fairing of parametric patches, respectively.

Implicit The surface is a level set surface, and is defined to pass through all positions where the implicit function evaluates to some specified value (usually zero). Least-squares fitting to an implicit line equation is reviewed in section 2. The global algebraic representations used by Taubin and Gotsman are covered in section 3. The piecewise algebraic implicit functions of Bajaj are covered in section 4.

Simplicial In this representation, the surface is a collection of simplicial complexes including points, edges, and triangles. Techniques to identify which simplicial complex belongs to the surface include Alpha shapes and the Crusts algorithm. Both of these topics are covered briefly in this paper.

2 Least-Squares Fitting

We begin with a description of least-squares fitting, one of the simplest and most frequently used techniques for curve and surface reconstruction. This technique is especially applicable to piecewise polynomial fitting in that the coefficients for each polynomial is often determined by least-squares fiting. Least-squares finds the coefficients for the curve or surface which minimize the squared distance between the curve or surface and the data points. It is one method of minimal distance fitting. The distance used in minimal distance fitting may be the orthogonal distance with respect to the curve or surface, or it may be aligned to an arbitrary axis. For example, given a one dimensional curve in Euclidean space, $y = f(x)$, the distance from a data point, (x_i, y_i), to the curve along the y-axis is simply the difference between y_i and $f(x_i)$. This distance is not, however, the orthogonal distance since the curve may have a non-zero slope. To show the difference in fitting using these two distance metrics, we take the concrete example of fitting a set of 2D data points to an implicit line, given by $Ax + By + C = 0$. To avoid a trivial solution of (0,0,0) for A, B, and C, we divide by B and rearrange the line equation to the following:

$$y = C_1 x + C_2 \tag{1}$$

The line equation can be applied to each data point, (x_i, y_i) to obtain the following linear system:

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ ... & 1 \\ x_n & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ ... \\ y_n \end{bmatrix} \tag{2}$$

n is the number of data points. The coefficients, C_1 and C_2, are found by solving the over-constrained system using Singular Value Decomposition. By rearranging the line equation to avoid the trivial solution, we have made the assumption that B is non-zero. The rearrangement also results in distance measured along the y-axis.

In least-squares fitting, the squared *orthogonal* distance is minimized. In this case, we do not rearrange the line equation. Instead, we can directly calculate the distance, d, of a data point to the implicit line by evaluating the line equation, $d = Ax + By + C$. The squared distance is:

$$D = d^2 = (Ax + By + C)^2 \tag{3}$$

To minimize, D, the derivative of (3) is taken with respect to A, B, and C and set to zero. The resulting equations are a linear system which can be formulated as a matrix equation:

$$2 \begin{bmatrix} x^2 & xy & y \\ xy & y^2 & y \\ x & y & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

The matrix on the left is the outer product of a single data point, $(x_i, y_i)(x_i, y_i)^t$. The above system is applied to all the data points by taking the summation of the outer products of all the points. Eigenvector decomposition is performed on the resulting 3x3 matrix to obtain the null space of the system. The null space is the solution because it is exactly the vector space in which the least-squares distance is zero. In this example, the vector space is the coefficients of the line we wish to reconstruct. In particular, the eigenvector corresponding to the smallest eigenvalue is the solution to the unknown coefficients.

3 Global Algebraic Surfaces

In the global algebraic scheme of surface reconstruction, the goal is to fit a trivariate polynomial to a set of constraints. The resulting surface is the locus of points for which the polynomial evaluates to zero. The degree of the polynomial is generally a user-specified parameter. Given a polynomial of degree n, the reconstruction algorithm finds the coefficients for each term in the polynomial that results in the best fit to the data, based

on a distance metric which may be the Euclidean distance. The coefficients may be found through steepest descent or some other optimization technique.

These algebraic surfaces are global in that only one polynomial is used to represent the entire surface. This global aspect results in several drawbacks to these techniques. In a one dimensional height field, the degree of the polynomial determines the number of inflection points in the resulting curve. A straight line has no inflections and is represented by a first degree polynomial. Parabolas and hyperbolas have one inflection point and are represented by second degree polynomials. Third degree (cubic) polynomials are needed to represent curves with three inflection points. In general, higher degree polynomials are needed to represent curves and surfaces with a large number of inflection points. In three dimensions, higher degree polynomials correspond to highly varying surfaces and complex topology. With higher degrees, however, there are many more coefficients for which to solve, resulting in a higher possibility of being caught in a local minimum during optimization. Incorrect coefficients would be obtained that may fit the data well but does not represent the desired surface well. As a result, there is a conflict between using a polynomial with a degree high enough to represent all the detail of a surface such as a bunny which has many inflection points, and one that has few enough coefficients to be tractable. A second drawback of global algebraic surfaces is that the user generally needs to select the polynomial (and thus the highest degree) to use. This selection is non-intuitive, and as in most cases, experience of the algorithm or trial and error is necessary to obtain a desirable surface.

Two examples of global algebraic surface reconstruction techniques include the works of Gabriel Taubin and Craig Gotsman.

3.1 Taubin's Polynomial Fitting

Taubin fitted polynomial functions to two dimensional closed curves and three dimensional surfaces. In [10] and [11], he uses the Levenberg-Marquardt algorithms as the optimization technique to search for the coefficients of the polynomials. In [10], Taubin formulates the objective function to be minimized and shows how the optimization becomes a generalized eigenvalue problem. In [11], he improves the objective function by defining a new approximate distance metric.

The surface that best fits a set of data points is often defined to be the surface that minimizes the Euclidean distance to all the data points. However, as Taubin points out, measuring Euclidean distances to implicit functions requires an iterative process. Implicit functions are not often Euclidean distance functions. Their evaluation at an arbitrary point indicates, by the sign of the value, whether the point is on, inside, or outside of the surface. The value itself may be an indicator of closeness to the surface but it is not the point's actual Euclidean distance to the surface. The evaluation cannot even be considered equal to the Euclidean distance by a scale factor because the implicit distance may be nonlinear. As a result, it is necessary to formulate an approximation to the Euclidean distance.

In [10], Taubin uses a first order approximation of the Euclidean distance:

$$Dist(x, Z(f))^2 \approx \frac{f(x)^2}{\| \nabla f(x) \|^2} \tag{5}$$

$Z(f)$ is the set of zeros of the function, $f(x)$, representing the surface. x is a data point. $\nabla f(x)$ is the derivative of $f(x)$ with respect to point x and supplies the direction and magnitude towards the surface from point x. It indicates the location of the surface, and the distance traveled in function space per unit traveled in point space. Intuitively, dividing the distance in function space to reach the surface $(f(x) - f(surface))$ by $\nabla f(x)$ gives an approximate Euclidean distance to the surface if the function, $f(x)$ is roughly linear near x. In this case, $f(surface) = 0$ since the function evaluates to zero on the surface. In order to minimize the distance over the entire set of data points, the approximate distances are combined into a mean approximate distance:

$$MeanD = \frac{1}{q} \sum_{i=1}^{q} \frac{f(p_i)^2}{\| \nabla f(p_i) \|^2} = \frac{\frac{1}{q} \sum_{i=1}^{q} f(p_i)^2}{\frac{1}{q} \| \nabla f(p_i) \|^2} = \frac{FMF^t}{FNF^t} \tag{6}$$

q is the number of data points. p_i is a specific data point. The mean approximate distance is obtained

by summing the Euclidean distances from the surface of all data points and dividing by the number of data points. The mean approximate distance is simplified to the final form on the right. F is a vector of the coefficients of the polynomial function, $f(x)$. M and N are matrixes containing the following terms:

$$M = \frac{1}{q}\sum_{i=1}^{q}[X(p_i)X(p_i)^t] \tag{7}$$

$$N = \frac{1}{q}\sum_{i=1}^{q}[DX(p_i)DX(p_i)^t] \tag{8}$$

X is a two or three dimensional vector for each data point. Each product, $X(p_i)X(p_i)^t$, is the 2x2 or 3x3 outer product matrix. DX is the Jacobian matrix of X. The gradient of the function is evaluated at each data point and stored in matrix form in the Jacobian. The entries of M and N are linear combinations of the moments of the data points. The minimizer of the above equation is the eigenvector corresponding to the minimum eigenvalue of the pencil $F(M - \lambda N) = 0$. An analysis on the generalized eigenvector fit can be found in the appendix of [10]. A higher order approximation of the Euclidean distance is presented in [11].

Taubin's results include both two and three dimensional curves and surfaces. He fitted a sixth order polynomial to an image of a pair of pliers. Three dimensional surfaces were reconstructed from synthetic data of a bean, a cup, and a torus, and range and CT data of a tooth. Taubin used fourth order polynomials for the surface fits.

3.2 Gotsman and Keren Polynomial Fitting

Gotsman and Keren's approach is to create parameterized families of polynomials that satisfy certain good properties, such as a tight fit. Previous techniques use a cost or energy function that penalizes improper fits defined by the distance from the data. The aim of this work is to find an analytical parameterization of a sub-family of polynomials that already satisfy desirable properties. This family must be as large as possible so that it can include as many functions as possible. This technique leads to an over-representation of the subset, in that the resulting polynomial will often have more coefficients to solve for, requiring additional computation. However, the family of larger polynomials will yield better results than the original family of polynomials with which we start [8].

Gotsman and Keren apply their approach to both two dimensional and three dimensional curves and surfaces. Two families of parameterizations are presented - starshaped zero sets and convex planar polygons. The second parameterization is also extended to 3D convex polyhedrons. We will now discuss the starshaped parameterization in the next section. The same technique is used to derive parameterizations for convex planar polygons and polyhedrons. Hence, derivations for the polygon and polyhedron will not be discussed in this paper.

3.2.1 Starshaped Zero Sets

In 2 dimensions, the starshaped family of polynomials prevents pathologies such as holes, loops, folds, and extraneous components because a starshape requires the existence of an interior point from which the entire curve is visible. This point is called the kernel point. For simplicity, the kernel point is assumed to be at the origin (translations of the object can make this possible).

Gotsman and Keren use a fourth order polynomial as the base function in their initial examples. As common among global algebraic surfaces, the selection of a base polynomial is non-intuitive and is based on experience with the different classes of families. The fourth order function is:

$$\begin{aligned} P(x,y) \;=\; & a_{40}x^4 + a_{31}x^3y + a_{22}x^2y^2 + a_{13}xy^3 + a_{04}y^4 + \\ & a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + a_{03}y^3 + \\ & a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{10}x + a_{01}y + a_{00} \end{aligned} \tag{9}$$

The above polynomial can be changed to a function in x by constraining y to lie on a line through the origin. These lines are of the form $y = \alpha x$ since the y-intercept is zero. Equation (5) is reformulated as follows:

$$\begin{aligned} P_\alpha(x) &= (a_{40} + a_{31}\alpha + a_{22}\alpha^2 + a_{13}\alpha^3 + a_{04}\alpha^4)x^4 + \\ &\quad (a_{30} + a_{21}\alpha + a_{12}\alpha^2 + a_{03}\alpha^3)x^3 + \\ &\quad (a_{20} + a_{11}\alpha + a_{02}\alpha^2)x^2 + \\ &\quad (a_{10} + a_{01}\alpha)x + a_{00} \end{aligned} \tag{10}$$

This function has 15 degrees of freedom. Roll's theorem is used to limit this polynomial to a starshaped zero set. In summary, Roll's theorem says that if a line through the origin intersects the zero set at more than two points (violating the starshaped constraint), then the second derivative of the polynomial with respect to x will have at least one root. To ensure that a line through the origin intersects at no more than two points on the starshaped surface, the second derivative is required to be positive for every x and α. The second derivative with respect to x is of the form:

$$\begin{aligned} &(a_{40} + a_{31}\alpha + a_{22}\alpha^2 + a_{13}\alpha^3 + a_{04}\alpha^4)x^2 + \\ &\quad (a_{30} + a_{21}\alpha + a_{12}\alpha^2 + a_{03}\alpha^3)x + \\ &\quad (a_{20} + a_{11}\alpha + a_{02}\alpha^2) \end{aligned} \tag{11}$$

Let the above class be denoted by *POS*. As mentioned above, *POS* needs to be everywhere positive. Polynomials that are everywhere positive can be generated by summing the squares of other polynomials. Additional classes of polynomials are defined below:

ROOT is a polynomial which is squared to obtain elements of *POS*. An example of *ROOT* is given:

$$L_{21}\alpha^2 x + L_{20}\alpha^2 + L_{11}\alpha x + L_{02}x^2 + L_{10}\alpha + L_{01}x + L_{00} \tag{12}$$

SUMSQ is a subset of polynomials in *POS* which are sums of squares of polynomials in *ROOT*.

Given these sets of families, the following questions arise:

1. Are *SUMSQ* and *POS* identical?
   No. There are everywhere positive polynomials which cannot be represented as sums of squares.
2. If $SUMSQ \neq POS$, does *SUMSQ* have a full dimension of 15 so that no degrees of freedom are lost?
   The dimension of *POS* is equal to the number of terms. The dimension of *SUMSQ* is equal to the number of monomials that can be expressed as an average of two even monomials. For example, the monomial $a_{30}x\omega^5$ is denoted by (0,1,5) which are the degrees of (α, x, ω). It is the average of (0,2,4) and (0,0,6) which are the monomials $a_{40}x^2\omega^4$ and $a_{20}\omega^6$, respectively. Note that ω is used to normalize all the terms of *POS* to be sixth degree monomials.
3. What is the minimal number of elements of *ROOT* which must be squared and summed in order to obtain all the elements of *SUMSQ*?
   We want to sum as few as possible without losing any degrees of freedoom.

The Pythagoras number defines the lower bound on the number of squares which must be summed in order to obtain every element of *SUMSQ*.

Using the above properties, we can obtain the elements of *ROOT* that must be squared and summed to guarantee that *SUMSQ* is covered. This set of elements will often have more parameters (unknown coefficients) than the original polynomial (the fourth order polynomial). In the example in [8], there were 30 parameters. The polynomial for the desired curve is obtained by taking the integral of the sum of squared elements of *ROOT* (the sum of squared elements is *SUMSQ* or *POS*). Recall that *SUMSQ* or *POS* are second derivatives of the polynomial curve. Linear and constant terms were lost when the second derivative was taken. Hence, additional linear coefficients and constants need to be added to the polynomial. The coefficients of the polynomial curve are solved using the Levenberg-Marquardt (LM) optimization algorithm.

Optimization is actually performed on the coefficients of $ROOT$, not the resulting polynomial curve. We need to optimize on $ROOT$ because we need to ensure that the reconstructed surface has the desirable properties. If optimization were performed on the resulting polynomial curve, then the coefficients may become any arbitrary value (so long as the resulting surface fits the data), and they may break the desirable properties we want to maintain. By optimizing on the $ROOT$ coefficients, we are guaranteed that POS will still be a positive polynomial since $ROOT$ is squared. The LM algorithm requires initial coefficients. These coefficients are then modified at every time step. Each time that the coefficients are changed, POS and the polynomial curve are generated. The distance of the data points to the polynomial serves as the error, or measure of goodness of fit. LM takes a steepest descent path to the lowest error using the derivative of the error with respect to the coefficients. In summary, the coefficients of the polynomial curve are solved by repeatedly changing the coefficients of $ROOT$, generating POS, obtaining the polynomial curve, and then calculating the error and derivative.

Results for the star-shaped family of family of polynomials include a fourth order polynomial fit to an eye and a sixth order polynomial fit to a convex pentagon. Three dimensional examples include a sixth order polynomial fit to a cube and a house (nine-sided polyhedron).

The primary drawbacks of this technique for surface reconstruction are that the user must select a base class of polynomials, and the technique is not scalable. As previously mentioned, selection of a base polynomial with which to fit the data is not an intuitive process. The technique is not scalable in that each base class requires a separate derivation. Examples were given for quartic, quintics, sextic (fourth, fifth, sixth degree) polynomials. However, if a seventh degree polynomial is used as the base family, then $ROOT$ and $SUMSQ$ must be derived specifically for the seventh degree polynomial.

4 Piecewise Algebraic Surfaces

Chandrajit Bajaj and his co-authors approach the problem of reconstruction of algebraic surfaces using a divide and conquer algorithm, creating piecewise algebraic patches (called A-patches) that combine to form one surface. The technique groups surface data points into triangles along the contour in 2D or tetrahedrons along the surface in 3D. No assumption of connectivity between the data points is made. [2] deals with 2D contours, while [3] deals with 3D surface patches. In both cases, the Bernstein-Bezier (BB) basis is used for each triangle or tetrahedron. Continuity can be preserved between the patches or curves. In the 2 dimensions, C^2 and C^3 are preserved. In the 3 dimensions, C^1 (tangent plane) is preserved. Each triangle or tetrahedron is divided into a series of smaller interior triangles or tetrahedra. The vertices and interior points of the triangle and the tetrahedron form the control points of the curve or patch. There are nine control points for a triangle, and twenty for a tetrahedron. The triangle (or tetrahedron) is the convex hull of the control points. Figure 1 shows the control points of a triangular patch. There are two primary steps in the algorithm: 1) splitting the data set into triangles or tetrahedrons, and 2) solving for the coefficients of the control points of each triangle or tetrahedron. [3] provides a better introduction on BB forms than [2]. In 3 dimensions, any polynomial in (x, y, z) of degree n can be expressed in BB form as:

$$f(p) = \sum_{|\lambda|=n} b_\lambda B^n_\lambda(\alpha) \tag{13}$$

$B^n_\lambda(\alpha)$ is the Bernstein polynomial and b_λ's are the coefficients of the control points.

$$B^n_\lambda(\alpha) = \left(\frac{n!}{\lambda_1!\lambda_2!\lambda_3!\lambda_4!}\right)(\alpha_1^{\lambda_1}\alpha_2^{\lambda_2}\alpha_3^{\lambda_3}\alpha_4^{\lambda_4}) \tag{14}$$

α's are the barycentric coordinates of a data point, p, and can be found using the cartesian coordinates of p ($[xyz1]$). λ's are defined by the subscript of the control points. We are solving for b_λ, the coefficients of the control points. b_λ can be found using a series of rules defined by the continuity that we wish to preserve and other conditions such as smooth vertices and smooth edges conditions which constrain certain coefficients to be non-zero. The reader should consult [3] for explanations and derivations of the rules. In general, the higher the continuity we wish to preserve, the fewer the degrees of freedom will be left, since certain control points will be negative and other positive depending on the continuity requirement. Figure 2 shows a comparison

between the control point constraints and the continuity condition that is preserved in the 2 dimensional case for a triangle.

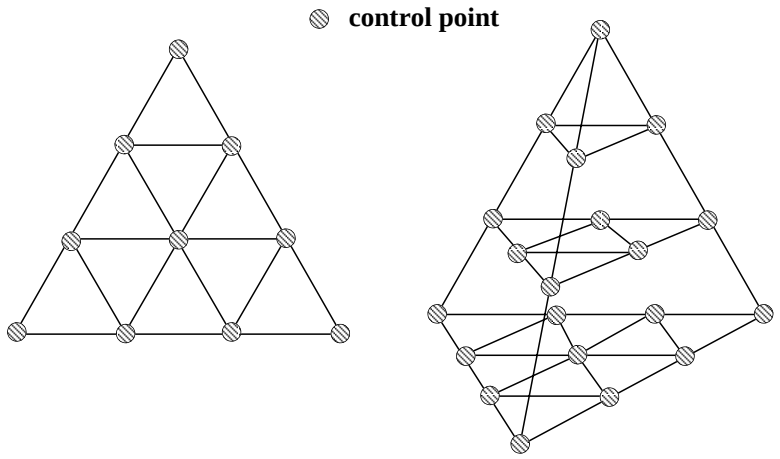

Fig. 1. Bernstein Bezier Coefficients of a C^0 Cubic Algebraic Curve.

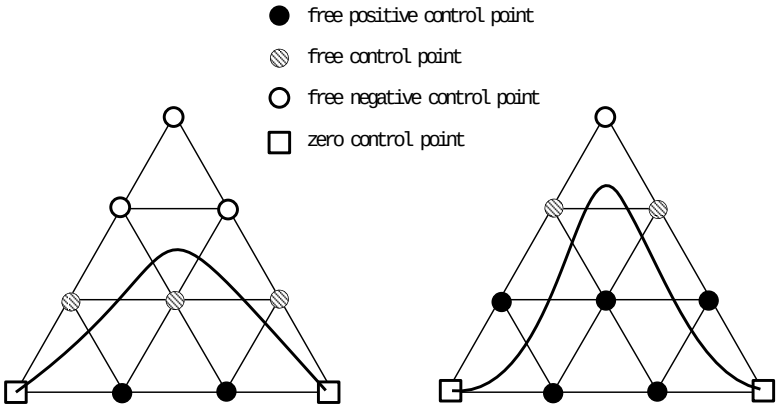

Fig. 2. Control points for a tetrahedron.

4.1 Generating Triangles for 2D Data

Given a set of 2D data points along the contour of an object, the data points must be grouped into sets. One cubic curve is generated per group, while preserving continuity (up to C^3) between curves (or groups). The data points are grouped according to the direction of their normal vectors. A circle is divided into k pie wedges. All consecutive data points of the 2D contour with normal vectors that fall into the same pie wedge are grouped together into one curve. Each curve exactly interpolates the endpoints and locally computed derivatives, and approximates the points interior to the curve using least squares. If a curve yields a large error when compared to the data points, additional cubic curves can be added by locally refining single pie wedges, so that smaller curves are generated. Normal vectors and derivatives are locally computed by a technique similar to forward differencing.

4.2 Generating Tetrahedrons for 3D Data

In the 3D case, we begin with a surface triangulation or mesh. For each face in the triangulation, normal vectors are calculated. The algorithm then distinguishes between convex and non-convex faces. One tetrahedron is generated for a convex face, and two tetrahedra are generated for non-convex faces. The same is done for edges.

4.3 Improved Algorithm for A-Patches

In [4], the same A-patch formulation is used. The major improvements in this paper are the technique by which the data set is divided into tetrahedrons, simplification of some of the rules that guide creation of three or four-sided A-patches, and the use of a three dimensional Clough-Tocher scheme that smooths out the surface by achieving C^1 continuity. The algorithm consists of four steps - classification of surface tetrahedra, computation of approximate signed distance function for each data point, creation of three and four sided patches for each tetrahedron, and surface smoothing using the Clough-Tocher scheme.

The data set is divided into tetrahedrons using incremental Delaunay triangulation. The incremental algorithm allows addition of arbitrary data points for refinement. Alpha shapes is used to filter out the simplices to obtain an adequately correct approximation of the object. Then, only the tetrahedra that contain the surface are kept to be processed. This classification is performed by traversing adjacent tetrahedra, starting with a tetrahedron that is external to the object. All external tetrahedra are marked as external and queued for traversal. If an internal tetrahedron is encountered it is marked as internal, but it is not enqueued for traversal. Once the queue is empty, all external tetrahedra have been marked as external and all boundary tetrahedra have been marked as internal. All other internal, but non-bounding, tetrahedra have not been marked or traversed. For each boundary tetrahedron, an A-patche is created using the BB formulation described in section 4.0.

One important addition in this paper is the use of the Voronoi duality to create a signed-distance function for the data points. The Voronoi is a dual of the Delaunay triangulation, and the entire region within a Voronoi cell is closest to the vertex of the Delaunay triangulation that resides in that cell. Given any data point, the location of the point within the triangulation is found, and its distance to the closest vertex is computed and used as the signed distance. The signed distance values are used as additional constraints in solving for the coefficients of the A-patches.

In the third step, the coefficients of the patches are solved as a least squares problem that is actually over constrained. For each tetrahedron, the polynomial must evaluate to zero at every data point in the tetrahedron, and there is a signed distance constraint associated with every control point. Once coefficients have been obtained for all the patches, an error measure is calculated based on the distance between the data point and the patches. If the error is above the threshold the surface is refined by finding the tetrahedron containing the largest error. The circumcenter of the selected tetrahedron is then added to the Delaunay triangulation (the Delaunay triangulation is maintained by the incremental approach). For each new tetrahedron, a patch is formed and the coefficients are solved as before.

Once the error is below the threshold, the above incremental refinement step may end. However, the surface of patches at this step is not C^1 continuous. The final step uses the Clough-Tocher scheme to make the patches C^1 continuous. Each tetrahedron is split into twelve tetrahedra. The coefficients of the twelve patches are computed based on the value of the function at each vertex of the new patches, the average gradient at the vertices and the mid-edge points, as well as the continuity constraints imposed between patches. Using these additional gradient constraints, C^1 continuity can be preserved between the patches.

An important aspect of this algorithm is the single-sheeted property that is maintained for every tetrahedron. The single-sheeted property prevents the polynomial within a tetrahedron to fold over onto itself. Figure 3 shows the two types of single-sheeted patches - three and four sided patches - with its corresponding control points and constraints. The reason for maintaining this property is not explicitly stated in [4]. Cubic polynomials can fold over itself up to two or three times. Most likely, the single-sheeted property needs to be maintained to ensure that the least squares fitting is well-behaved. This aspect of the algorithm reveals its drawback. The primary advantage of using polynomials to reconstruct surfaces is that they can well represent the curved aspects of the surface such as inflections in the surface. However, by constraining the polynomial in each tetrahedron to be single-sheeted, the reconstructed patch is fairly planar.

5 Thin-plate Parametric Surfaces

Terzopoulos present a reconstruction technique to generate three dimensional surfaces from depth images. Unlike the previous techniques covered in sections 2.0 - 4.0, this work is in the domain of parametric sur-

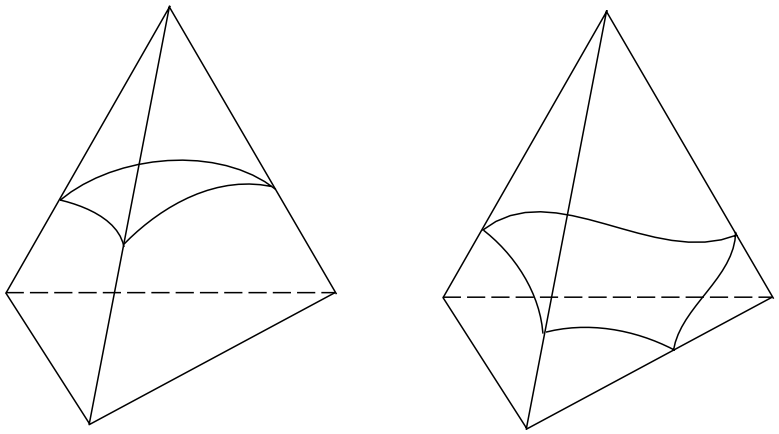

Fig. 3. Examples of three and four sided patches.

faces. Terzopoulos' goal was to create a smooth surface that also correctly represents the discontinuities and boundaries found in range images. These discontinuities are from the silhouette of objects as seen from one viewpoint. The two key notions of his approach are the use of an energy functional which controls continuity and accuracy of fit and the use of molecular configurations as masks in finite differencing. In [12], he poses the problem of reconstruction as an inverse problem with infinitely many feasible solutions. Adding constraints to fix upon one solution is the process of regularization. Terzopoulos' approach to regularization is the minimization of a functional which measures the smoothness of the surface function and the accuracy of fit to the data. The energy functional consists of two terms:

$$E(v) = K(v) + P(v) \tag{15}$$

v is the reconstructed surface function. $K(v)$ controls the continuity of the surface, and $P(v)$ is a penalty function for accuracy of fit. $K(v)$ includes a membrane and a thin-plate term:

$$K(v) = \frac{1}{2}\int\int_{\Omega} \rho(x,y)\{\tau(x,y)(v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2) + [1-\tau(x,y)](v_x^2 + v_y^2)\}\ dx\ dy \tag{16}$$

Ω denotes the image region. The first term $(v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2)$ is the thin-plate term and guarantees C^1 continuity of the surface. The second term is the membrane term, guaranteeing C^0 continuity. A thin-plate surface acts as a sheet of metal bent over the data points, while a membrane surfaces behaves more similarly to a balloon stretched over the data points. The thin-plate surface is stiffer and interpolates smoothly between two data points, while the membrane produces spike-like points where it interpolates the data. The two functions, ρ and τ, are weighting functions defined by the continuity found in the range images and by the desired local smoothness. As $\rho \Rightarrow 0$, the surface becomes locally discontinuous. As $\tau \Rightarrow 1$, the surface becomes a thin-plate spline, and as $\tau \Rightarrow 0$, the surface becomes a membrane. For intermediate values of ρ and τ, the surface may characterize a thin-plate surface under tension. Terzopoulos characterizes r as a spatially varying cohesion factor, and t as a spatially varying tension (or smoothness) factor.

The second term of the energy functional controls the accuracy of fit:

$$P(v) = \frac{1}{2}\sum_i \alpha_i (v(x_i,y_i) - d_{x_i,y_i})^2 \tag{17}$$

$v(x_i,y_i)$ are the reconstructed surface points, and d_{x_i,y_i} are the data points. The squared term is the Euclidean distance. α is a weighting factor determined by the confidence in the measured data point. If derivatives at each point data is given with the data set, additional stiffness terms may be added to the to the penalty function:

$$P(v) \quad = \quad \frac{1}{2}\sum_i \alpha_{d_i} (v(x_i,y_i) - d_{x_i,y_i})^2 +$$

$$\frac{1}{2}\sum_i \alpha_{p_i}(v_x(x_i,y_i) - p_{x_i,y_i})^2 + \frac{1}{2}\sum_i \alpha_{q_i}(v_y(x_i,y_i) - q_{x_i,y_i})^2 \quad (18)$$

$v_x(x_i,y_i)$ and $v_y(x_i,y_i)$ are the partial derivatives in the x and y directions at the reconstructed surface point, $v(x_i,y_i)$. p and q are the known partial derivatives in the x and y directions, respectively. α's are the weighting factors for accuracy of position and derivatives. Terzopoulos identify α_p and α_q as the spring stiffness.

Working in the image domain, Terzopoulos approximates the continuous energy functionals using discrete finite differencing. Finite differencing requires a grid of data points, which in this case, is the grid of pixels in a depth image. Each grid point is called a node. The discrete first and second partial derivatives at node $(i,j)$are as follows:

$$v_x^h = \frac{1}{h}(v_{i+1,j}^h - v_{i,j}^h) \quad (19)$$

$$v_{xx}^h = \frac{1}{h^2}(v_{i+1,j}^h - 2v_{i,j}^h + v_{i-1,j}^h) \quad (20)$$

This approach can be multi-scale in that each grid node may not correspond to a single pixel in the image but a neighborhood of pixels. h corresponds to the element size. In order to minimize the discrete energy functional, the spatially varying cohesion and tension factors, ρ and τ, are fixed. The derivative of the energy with respect to each nodal position is set to zero:

$$\frac{\partial K_{\rho\tau}^h(v^h)}{\partial v_{i,j}^h} + \frac{\partial P^h(v^h)}{\partial v_{i,j}^h} = 0 \quad (21)$$

Expanding the continuity and penalty terms in the above equation results in nodal equations which incorporate the discrete partial derivatives. From the nodal equations, Terzopoulos develops a series of molecule masks which encapsulate the depth, orientation, membrane, and thin-plate constraints at each node. Separate molecules for the constraints are summed together to form one molecule at each node. The molecules take on different configurations depending on the continuity that may be inhibited, corresponding to a boundary discontinuity. Each element in the molecule is applied to the image, and the total is set to zero, corresponding to the minimization of the discrete energy functional. The system of nodal equations is a sparse matrix because of the locality of the finite element representation, and it is symmetric. The size of the matrix depends on the image size which may be quite large. Given these properties, Terzopolous solves the system iteratively using relaxation methods.

The results show three dimensional surface reconstructions of a sphere, a torus, a light bulb, and terrain data. Several reconstructed surfaces include discontinuities such as steps and holes.

The primary disadvantage of the finite differencing technique is that the resulting surface is a discrete representation, defined by nodal positions. This is unlike the techniques discussed in the previous sections in which the surface is a polynomial or patches of polynomials. In addition, the reconstructed surfaces are topologically constrained to height fields.

6 Gossard Fairing of Parametric Surfaces

Celniker and Gossard present a three phase, interactive approach to modeling smooth parametric surfaces in [5]. The three phases consist of defining character lines of the shape, applying the deformable surface to the character lines, and applying loads to deform the surface. This work differs from that of Terzopoulos in that finite elements and fairing is used rather than finite differencing. The difference between finite elements and finite differencing is that finite differencing solutions generate positions for a set of vertices that form the mesh of a surface. Finite elements, on the other hand, result in a set of weights. These weights are applied to the pre-defined basis functions to form the surface. Both techniques minimize an energy functional in order

to solve for the unknowns. As it turns out, the energy functional used by Celniker and Gossard is similar to that used by Terzopoulos. Celniker and Gossard develop both two dimensional curve and three dimensional surface finite elements. There are two key sections to their work: development of the deformable surface element and fairing of the surface.

The deformable surface element consists of triangular primitives with C1 continuity between primitives. Three different shape basis are defined for each vertex and one at each edge midpoint, totaling twelve basis for each triangle primitive. The shape functions are based on those defined by Zienkiewicz in The Finite Element Method, and are in terms of barycentric coordinates of the vertices. The three shape basis for one vertex of a triangular primitive are repeated below:

$$\begin{aligned}
\varphi_1 &= L_1 + L_{12}L_2 + L_{12}L_3 - L_1L_{22} - L_1L_{32} \\
\varphi_2 &= c_3(L_{12}L_2 + 0.5L_1L_2L_3) - c_2(L_{12}L_3 + 0.5L_1L_2L_3) \\
\varphi_3 &= -b_3(L_{12}L_2 + 0.5L_1L_2L_3) + b_2(L_{12}L_3 + 0.5L_1L_2L_3)
\end{aligned} \tag{22}$$

φ_1, φ_2, φ_3 are the shape functions. L_1, L_2, L_3 are the barycentric coordinates of a vertex, and b_2, b_3, c_2, c_3 are part of the matrix to transform cartesian coordinates into barycentric coordinates:

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = \frac{1}{2\triangle} \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} 1 \\ u \\ v \end{bmatrix} \tag{23}$$

Shape functions for the remaining two vertices of the triangular primitive is generated simply by shifting all the indices of equation (18) as follows: $1 \Rightarrow 2$, $2 \Rightarrow 3$, and $3 \Rightarrow 1$.

In [5], partial derivatives with respect to the cartesian coordinates (u, v) are derived using barycentric coordinates. These equations provide a mapping from the cartesian to the barycentric coordinate system that is later used for fairing of the surface. The energy functional is expressed in (u, v) coordinates. It consists of a fairness term and a force due to the applied loads on the surface. The fairness term includes a stretch resistance and a bending resistance. The first derivative of the surface is used as the stretch resistance, corresponding to a membrane. The second derivative of surface is used as the bending resistance, corresponding to a thin-plate. The applied loads are used to deform the surface. The energy functional is as follows:

$$\begin{aligned}
\int \{(\alpha_{11}|w_u|^2 + 2\alpha_{12}|w_u||w_v| + \alpha_{22}|w_v|^2 + \\
\beta_{11}|w_{uu}|^2 + 2\beta_{12}|w_{uv}| + \beta_{22}|w_{vv}|^2) - 2f \cdot w\}\, du\, dv
\end{aligned} \tag{24}$$

w is the deformable parametric surface. w_u, w_v, w_{uu}, w_{uv}, w_{vv} are the first and second derivatives with respect to the cartesian coordinates (u, v). α's and β's are the membrane and thin-plate weights. f is a vector of applied loads. As with Terzopoulos' surfaces, C^1 continuity is maintained by the thin plate energy term.

The overall algorithm is as follows. An initial set of shapes for each triangle primitive is built from the characteristic curves supplied by a user. These characteristic curves set certain vertices as constraints which cannot be deformed. The user may also apply loads to deform the surface. The unknowns to be solved during each iteration of the system are the positions and tangent vectors of the vertices that are not constrained. To solve for the unknown positions and tangent vectors, the energy defined above must be minimized. Celniker and Gossard use the Euler differential equations. To find the minimal energy, the differential equation is set to zero. The resulting relation for the surface balances the internal forces due to stretching and bending and the external forces due to applied loads:

$$\begin{aligned}
&\left(\frac{\partial^2(\beta_{11}w_{uu})}{\partial u^2} + \frac{\partial^2(\beta_{12}w_{uv})}{\partial u \partial v} + \frac{\partial^2(\beta_{22}w_{vv})}{\partial v^2}\right) - \\
&\left(\frac{\partial(\alpha_{11}w_u + \alpha_{12}w_v}{\partial u} + \frac{\partial(\alpha_{12}w_u + \alpha_{22}w_v}{\partial v}\right) = f
\end{aligned} \tag{25}$$

Using partial derivatives in terms of barycentric coordinates, the energy functional becomes a system of equations with the vertex positions and tangent vectors as unknowns. The system matrix is sparse when there are many triangular primitives since the shape basis are local to each triangle. The solution is found for energy equal to zero.

Celniker and Gossard's results include examples of interactive modeling. A wine glass is modeled from simple characteristic curves and applied loads. Examples of varying bending resistance and applying loads to simple constrained surfaces is also shown.

In [7], Fang and Gossard use the above approach to reconstruct smooth parametric surfaces from a cloud of points. An extra spring term that controls the accuracy of fit is added to the energy functional. The spring term is due to the spring force between the data points and the surface. A stiff spring forces the surface to exactly interpolate the data points, while a flexible one allows the surface to approximate the data points. The trade-off between fairness and accuracy of fit is controlled by a weighting parameter. The energy functional used for the surface is as follows:

$$\begin{gathered}\int\{(\alpha_{11}|w_u|^2+2\alpha_{12}|w_u||w_v|+\alpha_{22}|w_v|^2+\\ \beta_{11}|w_{uu}|^2+2\beta_{12}|w_{uv}|+\beta_{22}|w_{vv}|^2)-2f\cdot w\}\,du\,dv+\\ \tfrac{R}{2}\textstyle\sum_i K_i d_i^2\end{gathered} \tag{26}$$

The first term remains the same as in equation (2). In this case, f corresponds to the forces acting on the surface due to the sample points. R is the weighting of the springs' strain energy and controls the trade-off between fairness and accuracy. i goes from 1 to the number of data points. K_i is the spring constant for each spring associated with a data point, and d_i is the shortest distance between each data point and the surface.

The parametric surface is reconstructed by first fitting boundary curves to the boundary points. Repelling soap bubbles are placed on the surface and allowed to scatter according to the curvature of the surface. A Delaunay triangulation is then constructed from the centers of the soap bubbles.

Characteristic curves are added which constrain the points on the interior of the surface. The surface is then remeshed with the additional characteristic curves. The energy functional is minimized iteratively until convergence is reached. The convergence criterion is based on the desired fairness and accuracy of fit. Details on the minimization and the soap bubble scattering algorithms are not presented in [7]. Details can be found in [Fang 92 Visual Computing].

In order to attach the data points to the surface for the spring term, the minimum distance between each point and the surface must be found. On the first iteration, each point is compared against each surface triangle. As minimization progresses, the assumption is made that the surface shape will change slowly, so the data points may move only to neighboring polygons, so after the first iteration, it is no longer necessary to compare every point to every polygon in the mesh. Once attachments between data points and the surface have been established, the forces applied to the surface by each point must be resolved. Within each triangle, forces are resolved to the vertices by using the distance from each force point to each vertex as the weight. The force at each vertex is then weighted by area of the triangle in order to normalize the forces across the entire mesh. Results include fitting a surface to the data points of a car hood.

7 Simplicial Surfaces

Simplicial surfaces are not the main topic of this paper, but are briefly discussed in this section to complete the taxonomy of surface reconstruction algorithms. Simplicial surfaces are composed of points, edges, and triangles, and thus, are piecewise polynomial fits. A point is a zero dimensional simplex; an edge is a two dimensional simplex; and a triangle is a three dimensional simplex. The key notion behind simplicial complexes is the convex combination. An edge is the convex combination of two points, and a triangle is the convex combination of three points. The reconstruction techniques that generate simplicial surfaces construct simplicial elements from a collection of data points and then identify which simplicial complexes belong to the surface. Two such algorithms include Alpha Shapes introduced in [Edel 94] and the Crusts algorithm presented in [Amen 98]. The primary drawback to simplicial techniques is their reliance on the accuracy

of the data points. The vertices of the reconstructed surface are a subset of the original data points. Any noise attached to the data points by the method of data collection will directly translate to the reconstructed surface. Alpha Shapes and the Crusts algorithm are briefly discussed in the next two subsections.

7.1 Alpha Shapes

Edelsbrunner's Alpha Shapes technique consists of three steps - triangulation of the point set, selection of alpha radius, and identification of the simplicial complexes that are to be included in the reconstructed shape. The point set is triangulated using the Delaunay triangulation algorithm. This technique generates tetrahedrons, and the resulting shape is the convex hull of the point set. The point set is assumed to be in general position - no four points on a plane, no five points on a sphere. The special feature of a Delaunay triangulation is that for each tetrahedron, there is no other point, aside from the vertices of the tetrahedron, inside the circumscribing sphere of the tetrahedron. For each triangle and each edge, there is no other point inside the circumscribing circle. In the second step of the Alpha Shapes algorithm, the alpha radius used to carve out the alpha shape is selected. The process of carving out the shape is the final identification step, wherein simplicial elements are either kept or eleminated from the shape. The shape is carved out by removing edges, triangles, tetrahedrons whose circumscribing sphere is larger than the alpha ball. The original data points are never removed. At an infinitely large radius, no simplicial elements are removed, and the resulting shape is the convex hull. At an infinitely small radius, only the original point samples are left. The output of the Alpha Shapes technique is not a manifold surface, but a collection of simplicial complexes which approximate the 3D shape. The interior of the shape consists of tetrahedrons. The shape boundary may be considered the surface, though it is highly non-manifold since the shape boundary may contain disjoint or partially joined edges and triangles. Selection of bounding complexes to obtain a surface is a non-trivial task.

7.2 Crusts Algorithm

Amenta's Crusts algorithm, also called Voronoi filtering, consists of four steps - triangulation of the point set, addition of Voronoi vertices, retriangulation, and identification of the simplicial complexes that are to be included in the reconstructed surface. Delaunay triangulation is performed to generate the initial mesh. Next, the vertices of the Voronoi diagram (dual of the Delaunay triangulation), are added to the set of points to be triangulated. The Voronoi vertices are added because in two dimensions, they approximate the medial axis of the curve. Delaunay triangulation is applied to the union of the original set of points and the Voronoi vertices. The resulting triangulation delineates the edges that are on the boundary of the curve from those that are on the interior because interior edges have a Voronoi vertex as one of their vertices. Hence, the final step is to categorize edges as boundaries if their circumscribing circle is empty of all other sample points and of Voronoi vertices.

The Crusts algorithm does not perform as well in three dimensions because the Voronoi vertices do not well approximate the medial axis in three dimensions. Examples are cases in which the Voronoi vertices occur very close to the surface. To rectify this problem, not all the Voronoi vertices are added to the set to be retriangulated. Instead, for each sample point, only the two farthest Voronoi vertices of the sample point are included in the retriangulation. These two points are called the poles of the sample point. After triangulation of the sample points and selected Voronoi vertices, triangles are categorized as belonging to the surface of the shape only if all three vertices are sample points and not Voronoi vertices.

Another problem encountered by Amenti include disconnecting closely placed objects. An additional filtering step on surface normal vectors is necessary to eliminate abnormally oriented triangles which are often the result of the connection of two closely placed objects.

Note that one of the primary differences in this algorithm compared with the Alpha Shapes technique is that it reconstructs surfaces and boundaries, whereas Alpha Shapes reconstructs an entire object including the interior if there are interior data points. Interior data points are often present in medical data, such data from MRI or CT scans.

8 Conclusions

The universe of curve and surface reconstruction algorithms and representations is quite large. This paper attempts to survey several well-known techniques. The reconstructed curves and surfaces include parametric, implicit, and simplicial. More emphasis is given to implicit and algebraic representations, as developed by Taubin, Gotsman, Keren, Bajaj, Xu, Bernardini, and Chen. Representative papers from Terzopolous and Gossard were discussed as a comparison between finite differencing and finite elements. Simplicial techniques developed by Edelsbrunner and Amenti were briefly discussed.

Each algorithm has strengths and weaknesses for comparison. Global algebraic techniques tend to be elegant in constructing a single polynomial to represent a complex curve or surface. However, they rely heavily on the initial choice of the degree of the polynomial, and they are not easily scalable to highly detailed, topologically complex structures. Piecewise algebraic curves and surfaces attempt to solve the problem of scalability by dividing the curve or surface into multiple polynomials. As a result, they lose the power of representation capable of polynomials by constraining each polynomial in the piecewise representation to be fairly planar. The finite differencing solution introduced by Terzopolous is scalable and simplistic. Finite differencing and energy minimization is reduced to molecular masks which incorporate all the constraints at each element. This technique is, however, topologically constrained to height fields. The finite elements solution as developed by Gossard is more appropriately used in an interactive modeler, rather than as a reconstruction algorithm because loads need to be applied to deform the surface and the weighting between thin-plate and membrane needs adjustment. In [7], the technique is applied to surface reconstruction, but the single example of a car hood is fairly planar. Unlike many of the other approaches, simplicial techniques do not rely on minimization to fit a surface representation. However, they can only exactly interpolate the data and are thus sensitive to any noise present in the data.

Each representation is best suited to specific types of data. Simplicial techniques work well with dense data sets, such as medical data. The finite differencing solution is appropriate to image data which is uniform, dense, and can be represented by a height field. Global algebraic techniques work well for simple curves and surfaces because more complex surfaces require higher order polynomials, resulting in more coefficients for which to solve. The piecewise algebraic approach has the potential to handle topologically complex reconstructions, but again the data set must be dense since this technique relies on Delaunay triangulations and Voronoi diagrams.

References

[1] Amenta, N., M. Bern, and M.Kamvysselis. A New Voronoi-Based Surface Reconstruction Algorithm. *SIGGRAPH '98 Proceedings*, August, 1998.

[2] Bajaj, C. L. and G. Xu. Data Fitting with Cubic A-Splines. *Tech Report* , 1992.

[3] Bajaj, C. L., J. Chen, and G. Xu. Free Form Surface Design with A-Patches. *Graphics Interface Proceedings*, 1994.

[4] Bajaj, C. L., F. Bernardini, and G. Xu. Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans. *SIGGRAPH '95 Proceedings*, August, 1995.

[5] Celniker, G. and D. Gossard. Deformable Curve and Surface Finite-Elements for Free-Form Shape Design. *SIGGRAPH '91 Proceedings*, July, 1991.

[6] Edelsbrunner H. and E.P. Mucke. Three-Dimensional Alpha Shapes. *ACM Transactions on Graphics, Vol 13, No. 1*, January, 1994.

[7] Fang, L. and D. Gossard. Reconstruction of Smooth Parametric Surfaces from Unorganized Data Points. *Curves and Surfaces in Computer Vision and Graphics III, SPIE Vol. 1830* , 1992.

[8] Gotsman, C. and D. Keren. Tight Fitting of Convex Polyhedral Shapes. *Tech Report* .

[9] Keren, D. and C. Gotsman. Fitting Implicit Polynomials with Topological Constraints. *Tech Report* .

[10] Taubin, G. Estimation of Planar Curves, Surfaces, and Nonplanar Spaces Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 11*, November, 1991.

[11] Taubin, G. An Improved Algorithm for Algebraic Curve and Surface Fitting. *Proceedings Fourth International Conference on Computer Vision*, May, 1993.

[12] Terzopoulos, D. The Computation of Visible-Surface Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10, No. 4*, July, 1988.